**GIAO TIẾP CÁC MICROSERVICE QUA KAFKA**

**Tổng quát về Kafka Cluster**

**A diagram of a computer system

Description automatically generated**

* Kafka Cluster: Hệ thống phân tán bao gồm Apache Zookeeper và các Apache Kafka Brokers liên kết lại với nhau. ZooKeeper được sử dụng để quản lý và phối hợp các Kafka broker
* Mỗi Kafka Broker là một máy chủ độc lập hoặc một instance của Kafka đang chạy trong cluster
* Mô hình dữ liệu trong Kafka theo cấu trúc "broker - topic - partition - message". Broker chứa các topic, mỗi topic chứa nhiều partition, và mỗi partition chứa nhiều thông điệp được đánh số bằng offset

A diagram of a computer server

Description automatically generated

**Flow Kafka**

* Producer tạo thông điệp và gửi nó đến Kafka Broker thông qua một topic cụ thể
* Kafka Broker nhận thông điệp và ghi nó vào một trong các partition của topic
* Thông điệp có thể được sao chép (replicate) sang các broker khác để đảm bảo tính an toàn và dự phòng
* Consumer đăng ký để tiếp nhận thông điệp từ topic
* broker sử dụng offset để định tuyến thông điệp chưa được xử lý từ partition đến các consumer đã đăng ký
* Consumer nhận và xử lý thông điệp, và sau đó cập nhật offset để chỉ định vị trí đã xử lý
* Sau khi xử lý xong, consumer tiếp tục đọc và xử lý các thông điệp tiếp theo dựa trên offset đã được cập nhật

**Installation and Setup**

A black text on a white background

Description automatically generated

**Configuring Topics**

Chúng ta có thể tạo các chủ đề theo chương trình với AdminClient trong Kafka thay vì chạy command-line để tạo chủ đề trong Kafka như trước:

* Command-line:

A black and white text

Description automatically generated

* AdminClient: Chúng ta cần thêm bean KafkaAdmin, nó sẽ tự động thêm chủ đề cho tất cả các bean thuộc loại NewTopic:

A computer code with black text

Description automatically generated

**Producing Messages**

* Producer Configuration



A screenshot of a computer program

Description automatically generated

* Publishing Messages: Chúng ta sử dụng bean KafkaTemplate để gửi message

A computer code with red and black text

Description automatically generated

Khi gọi phương thức send, chúng ta có hai cách để xử lý kết quả:

* + Xử lý đồng bộ (synchronous): Điều này dẫn đến việc quá trình gửi tin nhắn phải đợi cho đến khi nhận được xác nhận từ Kafka Server rằng tin nhắn đã được gửi thành công và đã được ghi nhận => làm chậm producer => không phù hợp và không được khuyến khích

A screen shot of a computer code

Description automatically generated

A close up of a text

Description automatically generated

* + Xử lý bất đồng bộ (asynchronous): Sử dụng cơ chế callback của CompletableFuture để xử lý bất đồng bộ => cho phép producer tiếp tục xử lý các tác vụ khác mà không bị chặn

A screenshot of a computer code

Description automatically generated

A close up of a text

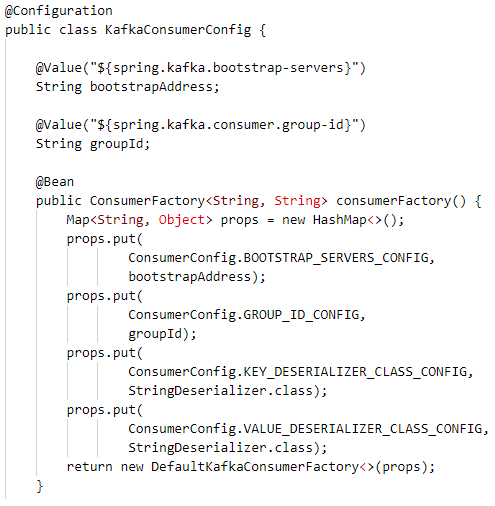
Description automatically generated

**Consuming Messages**

* Consumer Configuration:

A group of black text

Description automatically generated



A computer screen shot of a code

Description automatically generated

* Consuming Messages:

A white background with black text

Description automatically generated

**Document**

* https://www.baeldung.com/spring-kafka?fbclid=IwAR3AMouGlxl\_zqtCdrUns6rpMpWSW-0tqUTmGjMEpfBugt7n9LC0LbNmueg